Patrick Austin  
CS 482

Homework # 3

11/8/2017

2. To formulate the problem, we need an initial state and a goal. The initial state as shown in the diagram is:

On(B, Table) ^ On(C, A) ^ On(A, Table) ^ Clear(B) ^ Clear(C)

And the goal is:

On(A,B) ^ On(B,C)

Using the actions defined in Figure 10.3 of the textbook (p. 371) a plan that solves the problem is:

[ MoveToTable(C,A) , Move(B, Table, C) , Move(A, Table, B) ]

As specified in the problem, a noninterleaved planner is one that, when given subgoals to solve, produces a plan that is a concatenation of plans to solve the subgoals. Since the subgoals in this problem are On(A,B) and On(B,C), a noninterleaved planner can only produce a plan that puts A on B followed by a plan that puts B on C, or vice versa.

Say the planner attempted to get A on B, then attempted to get B on C. First the planner would clear C from B, then move A onto B, achieving On(A,B). Then the planner would then try to put B on C by clearing A from B, then moving C on to of B. Thus the planner achieves On(B,C) but in the process undoes the work done in the first step, leaving the problem unsolved. Similarly, concatenating the subgoal plans the other way around fails as well.

Interleaved planners, in contrast, allow for individual steps in the plans to solve the subgoals to be interleaved together rather than merely having the subgoal plans concatenated. This was the approach taken in the above solution: we intermixed steps from the subgoal plans sketched in the previous paragraph to create a complete solution. Since this interleaving is necessary to solve the problem, a noninterleaved planner cannot succeed.

3. We are given that a high-level action has one and only one implementation as a sequence of primitive actions. We need an algorithm for computing the high level action’s preconditions and effects, given that we know the complete refinement hierarchy and schemas for the primitive actions.

I will assume that we always choose the first possible primitive or refinement that we find when scanning the action from left to right, i.e. the leftmost one. It shouldn’t really matter what primitive or refinement we pick, since we have been told there is a unique solution, so always picking the leftmost should be fine.

The algorithm takes the high-level action as an input, and outputs a list of its preconditions and a list of its effects.

The algorithm should loop while the part of the high-level action that has not yet been processed is not empty (i.e. until all of its primitives and refinements, and refinements of refinements, and so on, have been processed).

The algorithm should remove the leftmost primitive or refinement from the high-level action for processing.

If this piece is a primitive, i.e. it can be refined no further, then its preconditions are put in the list of preconditions unless they are already in the list of effects. (If the precondition is already in the list of effects then it has already been met in the course of doing the high-level action so far and need not be considered in the final output, where the preconditions represent what is needed to trigger the whole, entire high-level action in the first place.) The effects of this primitive can simply be added to the list of effects, unless the effect to be added is the complement of an effect already in the list. In this case, where complementary literals occur, the new effect should be added to the list of effects, and its complement should be removed from the list. For example, if an effect of some primitive is ~SwitchOn, and we already have SwitchOn in the list of effects, SwitchOn should be removed from the list and ~SwitchOn should be added.

If this piece can be refined further, take the refinement that is achievable given the preconditions and effects we have acquired in the lists so far. Since there is a unique solution, this should always be possible and shouldn’t be ambiguous. Add this refinement’s preconditions unless they are already in the list of effects, and reattach the refined piece to the front of the remaining plan to be considered on the next loop of the algorithm. If it is another refinement it will be refined further, and if it is a literal it can be processed and removed on the next round.

In this fashion the algorithm will find the complete list of preconditions and effects for a high-level action with a unique implementation as a sequence of primitive actions.

4.

a. Since 5 cards are chosen from a 52 card deck with no repetition allowed, and order does not matter, the number of possible hands is (52 choose 5) = 2,598,960

b. Since each possible hand is unique and equally likely, the probability of each atomic event is 1 / (52 choose 5) = 3.85 x 10-7

c. There are 4 possible royal straight flush hands, one for each of the four suits. These constitute 4 possible hands out of the total, so the probability of getting a royal straight flush is 4 / (52 choose 5 ) = 1.54 x 10-6

There are 13 types of cards and thus 13 ways to get 4 of a kind. But since these are 5 card hands we also need to factor in the 48 other possible cards that can complete any given 4 of a kind. The number of hands is thus (13 \* 48) / (52 choose 5) = 2.4 x 10-4

5.

a. P(Toothache) constitutes all the probabilities on the left half of Figure 13.3. We can sum them to get the total P(Toothache), so P(Toothache) = .108 + .012 + .016 + .064 = .2

b. P(Cavity) constitutes all the probabilities on the top row of Figure 13.3. We can sum them to get the total P(Cavity), so P(Cavity) = .108 + .012 + .072 + .008 = .2

c. P(Toothache | cavity) constitutes the probabilities on the top row’s left half divided by those on the top row of Figure 13.3. Therefore

P(Toothache | cavity) = (.108 + .012) / (.108 + .012 + .072 + .008) = .6

d. P(toothache∨ catch) constitutes the probabilities the left half of Figure 13.3, as well as the probabilities covered by the catch column on the right half. We can sum them to get the total P(toothache∨ catch), so P(toothache∨ catch) = .108 + .012 + .016 + .064 + .072 + .144 = .416

P(Cavity | toothache∨ catch) is the values in the top row where toothache or catch are true divided by P(toothache∨ catch), found above. So

P(Cavity | toothache∨ catch) = (.108 + .012 + .072) / .416 = .462

6.

a. There are 64 possible outcomes. The expected payback can then be calculated as each outcome’s payoff times 1/64.

Since 1 outcome pays 20, 1 outcome pays 15, 1 outcome pays 5, 1 outcome pays 3, 3 outcomes pay 2, 12 outcomes pay 1, and all the other outcomes pay 0, we can calculate the expected payback as:

(20\*1)+(15\*1)+(5\*1)+(3\*1)+(3\*2)+(12\*1) / 64 = 61/64 = .953

In other words, playing one coin has an expected return of .953 coins.

b. If we define a win as receiving a non-zero payout from the machine, then the probability of one play resulting a win is simply all the non-zero-paying outcomes over the total number of outcomes, 64. Thus the probability of a win on a single play is:

1+1+1+1+3+12 / 64 = 19/64 = .297

7.

a. Network c claims this, since it asserts that Gfather , Gmother , and Gchild are all independent events with no links amongst themselves.

b. Networks a and b make claims that are consistent with the theory laid out in the problem description. Network a matches the theory most precisely, but network b may also match the theory if the extra links in network b compared to network a don’t actually assert dependence (i.e. they might be superfluous- we are free to include links in these networks that don’t actually “do” anything). Since network c says that the genes are all independent, it is not consistent with the theory.

c. Network a is the best description of the hypothesis, for the reasons mentioned in the previous part. Network b is only consistent when the links from Hmother and Hfather to Hchild are actually useless and contribute no dependence.

d. Attached.

e. Attached.

f. Since P(Gchild = l) = q + m - 2mq, as shown in part e., and we are assuming

P(Gchild = l) = P(Gmother = l) = P(Gfather = l) = q, we can solve for q (attached). On the attached page we show q must equal ½.

Since q = .5, we know this hypothesis does not match reality and must be mistaken. In reality, right handedness is much more common than left-handedness, whereas this hypothesis leads us to conclude that left-handedness and right-handedness are equally likely.